

Rec'd PTO

16 JUL 2003

PCT/JP03/16916

26.12.03

日本国特許庁
JAPAN PATENT OFFICE

JP03/16916

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日 2003年 1月 6日
Date of Application:

REC'D 19 FEB 2004

WIPO

PCT

出願番号 特願2003-000275
Application Number:
[ST. 10/C]: [JP2003-000275]

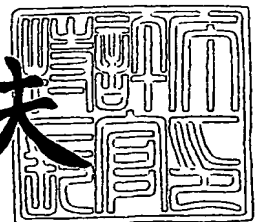
出願人 松下電器産業株式会社
Applicant(s):

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

2004年 2月 5日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫



出証番号 出証特2004-3006498

【書類名】 特許願

【整理番号】 2968140085

【提出日】 平成15年 1月 6日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 9/45

【発明者】

【住所又は居所】 愛知県名古屋市中区栄2丁目6番1号白川ビル別館5階
株式会社松下電器情報システム名古屋研究所内

【氏名】 河合 正樹

【発明者】

【住所又は居所】 愛知県名古屋市中区栄2丁目6番1号白川ビル別館5階
株式会社松下電器情報システム名古屋研究所内

【氏名】 川本 琢二

【発明者】

【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式
会社内

【氏名】 春名 修介

【発明者】

【住所又は居所】 大阪府門真市大字門真1006番地 松下電器産業株式
会社内

【氏名】 藤原 寛

【特許出願人】

【識別番号】 000005821

【氏名又は名称】 松下電器産業株式会社

【代理人】

【識別番号】 100097445

【弁理士】

【氏名又は名称】 岩橋 文雄

【選任した代理人】

【識別番号】 100103355

【弁理士】

【氏名又は名称】 坂口 智康

【選任した代理人】

【識別番号】 100109667

【弁理士】

【氏名又は名称】 内藤 浩樹

【手数料の表示】

【予納台帳番号】 011305

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9809938

【書類名】 明細書

【発明の名称】 コンパイラ、コンパイラを記録した記録媒体、コンパイル方法及びコンパイル装置

【特許請求の範囲】

【請求項 1】 原始プログラムを目的プログラムに変換するコンパイラであって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第 1 のサブプログラムから他プログラムモジュールに含まれる第 2 のサブプログラムを呼び出すステップを、

前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第 1 のサブプログラムから前記第 2 のサブプログラムへの移行ステップと、前記第 2 のサブプログラムから第 1 のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイラ。

【請求項 2】 原始プログラムを目的プログラムに変換するコンパイラであって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第 3 のサブプログラムから呼び出される第 4 のサブプログラムを含む他のプログラムモジュールの記述に基づいて、

前記原始プログラムに含まれる第 3 のサブプログラムから第 4 のサブプログラムを呼び出すステップを、

前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存する

ステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第3のサブプログラムから前記第4のサブプログラムへの移行ステップと、前記第4のサブプログラムから第3のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換し、

或いはまた、前記原始プログラムに含まれる第3のサブプログラムから第4のサブプログラムを呼び出すステップを、

前記第3のサブプログラムから前記第4のサブプログラムへの移行ステップを含む前記目的プログラムのステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイラ。

【請求項3】 前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第3のサブプログラムから呼び出される第4のサブプログラムを含む他のプログラムモジュールの記述は、

前記第4のサブプログラム毎に記述される宣言である、請求項2に記載のコンパイラ。

【請求項4】 原始プログラムを目的プログラムに変換するコンパイラであって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第5のサブプログラムから呼び出される第6のサブプログラムを含む他プログラム

モジュールに含まれる、呼び出された後のステップを、

前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、を含む前記目的プログラムのステップに変換する、コンパイラ。

【請求項5】 原始プログラムを目的プログラムに変換するコンパイラであって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第7のサブプログラムから呼び出される第8のサブプログラムを含む他プログラムモジュールの記述に基づいて、

前記原始プログラムに含まれる他プログラムモジュールに含まれる、呼び出された後のステップを、

前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、を含む前記目的プログラムのステップに変換し、

或いはまた、前記原始プログラムに含まれる他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイラ。

【請求項 6】 前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第 7 のサブプログラムから呼び出される第 8 のサブプログラムを含む他のプログラムモジュールの記述は、

前記第 8 のサブプログラム毎に記述される宣言である、請求項 5 に記載のコンパイラ。

【請求項 7】 前記呼び出される第 2 のサブプログラムまたは第 4 のサブプログラムを含む他プログラムモジュールを含む前記原始プログラムは、

前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域を使用するのか、

または、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの実行毎に固有にアクセスする前記他プログラムモジュール用の実行固有データメモリ領域を使用するのか、

に関する記述を含む、請求項 1 から請求項 3 のいずれか 1 項に記載のコンパイラ。

【請求項 8】 前記他プログラムモジュール用の実行共有データメモリ領域を使用するのかまたは前記他プログラムモジュール用の実行固有データメモリ領域を使用するのかに関する記述は、

前記他のプログラムモジュールに含まれる外部変数毎に記述される宣言である、請求項 7 に記載のコンパイラ。

【請求項 9】 前記原始プログラムに含まれる第 1 のサブプログラムまたは第 3 のサブプログラムから第 2 のサブプログラムまたは第 4 のサブプログラムを呼び出すステップを、

前記呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを保存するステップと、

前記呼び出される他のプログラムモジュール用の実行共有データメモリ領域アドレスを設定するステップと、

前記呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを保存するステップと、

前記呼び出される他プログラムモジュール用の実行固有データメモリ領域アドレスを設定するステップと、

前記第1のサブプログラムまたは第3のサブプログラムから前記第2のサブプログラムまたは第4のサブプログラムへの移行ステップと、

前記第2のサブプログラムまたは第4のサブプログラムから前記第1のサブプログラムまたは第3のサブプログラムへの復帰後に、

前記保存した呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して再設定するステップと、

前記保存した呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して再設定するステップと、

を含む目的プログラムのステップに変換することがあり、

前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行固有データメモリ領域をアクセスするステップを、

前記設定された他プログラムモジュール用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、

前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行共有データメモリ領域をアクセスするステップを、

前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、

を含む目的プログラムのステップに変換する、請求項7または請求項8に記載のコンパイラ。

【請求項10】 前記呼び出される第6のサブプログラムまたは第8のサブプログラムを含む他プログラムモジュールを含む前記原始プログラムは、

前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域を使用するのか、

または、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プ

プログラムモジュールの実行毎に固有にアクセスする前記他プログラムモジュール用の実行固有データメモリ領域を使用するのか、

に関する記述を含む、請求項 4 から請求項 6 のいずれか 1 項に記載のコンパイラ。

【請求項 11】 前記他プログラムモジュール用の実行共有データメモリ領域を使用するのかまたは前記他プログラムモジュール用の実行固有データメモリ領域を使用するのかに関する記述は、

前記他のプログラムモジュールに含まれる外部変数毎に記述される宣言である、請求項 10 に記載のコンパイラ。

【請求項 12】 前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、呼び出された後のステップを、

前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して設定するステップと、

前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して設定するステップに変換し、

前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行固有データメモリ領域をアクセスするステップを、

前記設定された他プログラム用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、

前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行共有データメモリ領域をアクセスするステップを、

前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、

を含む前記目的のプログラムのステップに変換する、請求項 10 または請求項 11 に記載のコンパイラ。

【請求項 13】 前記呼び出し元プログラムモジュール用の実行共有、および、実行固有データメモリ領域アドレスを保存するステップは、

前記第 1 のサブプログラムまたは第 3 のサブプログラムから第 2 のサブプログラムまたは第 4 のサブプログラムへ移行するために必要なデータよりも先に前記呼び出し元プログラムモジュール用の実行共有、および、実行固有データメモリ領域アドレスをスタックメモリに保存するステップを含む前記目的プログラムのステップである、請求項 1 から請求項 3 または請求項 7 から請求項 9 のいずれか 1 項に記載のコンパイラ。

【請求項 14】 前記原始プログラムに含まれる前記呼び出し元プログラムモジュールと、

前記呼び出し元プログラムモジュールに含まれる第 1 のサブプログラムまたは第 3 のサブプログラムまたは第 5 のサブプログラムまたは第 7 のサブプログラムから呼び出される第 2 のサブプログラムまたは第 4 のサブプログラムまたは第 6 のサブプログラムまたは第 8 のサブプログラムを含む他プログラムモジュールとは、同じプログラムモジュールである、請求項 1 から請求項 13 のいずれか 1 項に記載のコンパイラ。

【請求項 15】 コンピュータ読み取り可能な記録媒体であって、

請求項 1 から請求項 14 のいずれか 1 項に記載のコンパイラを記録した記録媒体。

【請求項 16】 請求項 1 から請求項 14 のいずれか 1 項に記載のコンパイラによって生成される目的プログラム。

【請求項 17】 コンピュータ読み取り可能な記録媒体であって、請求項 16 に記載の目的プログラムを記録した記録媒体。

【請求項 18】 原始プログラムを目的プログラムに変換するコンパイル方法であって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第 1 のサブプログラムから他プログラムモジュールに含まれる第 2 のサブプログラムを呼び出すステップを、

前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存する

ステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第1のサブプログラムから前記第2のサブプログラムへの移行ステップと、前記第2のサブプログラムから第1のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイル方法。

【請求項19】 原始プログラムを目的プログラムに変換するコンパイル装置であって、

前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムを呼び出すステップを、

前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第1のサブプログラムから前記第2のサブプログラムへの移行ステップと、前記第2のサブプログラムから第1のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、

前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、

前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイル装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本特許出願に係る発明（以後単に「本発明」とも言う）は、あるプログラム言語で記述された原始プログラム（以後、「ソースプログラム」または「ソースプログラムコード」または「ソースコード」とも言う）を、あるコンピュータによって実行するための目的プログラム「以後、オブジェクトプログラム」または「オブジェクトコードプログラム」または「オブジェクトコード」とも言う）に変換（以後場合によっては「翻訳」と言うこともあるが同一内容を示す）するプログラムであるコンパイラに関するものである。

【0002】

また本特許出願の明細書で言う、目的プログラムまたはオブジェクトプログラム等には、コンピュータが直接実行可能なプログラムコードに加え、コンピュータが直接実行可能なプログラムコードに変換する1段階である中間プログラムコードをも含むものであり、またそれに加え特に、コンピュータが直接実行可能なプログラムコードでは確定すべき実際にアクセスされるメモリアドレスが確定していない段階の中間コードプログラムをも含むものであり、更にまたそれに加え特に、コンピュータプログラムが直接実行可能なプログラムコード全体を一体化する以前の段階の中間コードプログラムをも含むものである。

【0003】

また例えば、いわゆる機器組み込みプログラムのようにオブジェクトプログラムを格納することのできるメモリ領域が限られており、しかも近年の携帯情報機器のようにその限られたメモリ領域に高度且つ高機能な情報処理のためのオブジェクトプログラムを格納しなければならないような目的に適う、オブジェクトプログラムを生成するためのコンパイラに関するものである。

【0004】

更にまた、このような目的に適うオブジェクトプログラムとして再入可能（以後、「リエントラント: *reentrant*」と言うこともある）なオブジェクトプログラムを自動生成するためのコンパイラに関するものである。

【0005】

【従来の技術】

従来、時分割または優先度プリエンプティブタスクまたは、プロセススケジューリング機能、及びそれに類する機能を有するオペレーティングシステムや、関数型プログラミング言語を用いることを条件として制作されるプログラムであって、複数のタスクまたはプロセスからある一定の時間間隔の中で、同時に呼び出される可能性がある関数プログラムは再入可能であることが要求された。

【0006】

そしてこのような再入可能関数プログラムは、人間のプログラマによってコーディングされ、再入可能性が満足されているか否かを自動的に判定するためのプログラムが作成されていた（例えば、特許文献1参照）。

【0007】

この再入可能性を満足する条件は、判定を行う対象である関数を記述しているソースプログラムの全体を構文解析してその全フローを抽出し、その全フローでアクセスされるすべてのリソースに対して、スコープを決定し、そのリソースのスコープの種類に応じて完全な排他的制御が行われているか否かに依存するものであった。

【0008】**【特許文献1】**

特開 2001-134431号公報

【0009】**【発明が解決しようとする課題】**

しかしながら、ソースプログラムの全体を構文解析してその全フローを抽出し、その全フローでアクセスされるすべてのリソースに対して、スコープを決定し、そのリソースのスコープの種類に応じて完全な排他的制御が行われているか否かを完全に行うことは時間的な制約やコンピュータの処理能力の限界から、現実には非常に困難で、ある一定レベルの判定に留まらざるを得なかった。

【0010】

そして従来の再入可能なプログラムの作成は、その本質的部分を人間のプログラマの作業によって記述されるため、多くの時間を要しながら不完全な排他的制

御しか実現することができず、その結果、プログラムを実行させた時に多くの誤動作を引き起こす原因となっていた。

【0011】

またこのような再入プログラムはマルチタスクや時分割環境下で使用されることが多いため、その原因発見は極めて困難であった。

【0012】

更にまた、このような再入可能なプログラムの更新や改訂を行うことは、更に一層困難なものであった。

【0013】

本特許出願に係る発明のコンパイラ、コンパイラを記録した記録媒体、プログラム生成方法及びプログラム生成装置は、これらの課題を解決するため、自動的に再入可能な目的プログラムを生成することを目的とするものである。

【0014】

【課題を解決するための手段】

上記課題を解決するため、本特許出願に係る請求項1に記載の発明は、原始プログラムを目的プログラムに変換するコンパイラであって、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムを呼び出すステップを、前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第1のサブプログラムから前記第2のサブプログラムへの移行ステップと、前記第2のサブプログラムから第1のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイラである。

【0015】

本特許出願に係る請求項2に記載の発明は、原始プログラムを目的プログラムに変換するコンパイラであって、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第3のサブプログラムから呼び出される第4のサブプログラムを含む他のプログラムモジュールの記述に基づいて、前記原始プログラムに含まれる第3のサブプログラムから第4のサブプログラムを呼び出すステップを、前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第3のサブプログラムから前記第4のサブプログラムへの移行ステップと、前記第4のサブプログラムから第3のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換し、或いはまた、前記原始プログラムに含まれる第3のサブプログラムから第4のサブプログラムを呼び出すステップを、前記第3のサブプログラムから前記第4のサブプログラムへの移行ステップを含む前記目的プログラムのステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイラである。

【0016】

本特許出願に係る請求項3に記載の発明は、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第3のサブプログラムから呼び出される第4のサブプログラムを含む他のプログラムモジュールの記述は、前記第4のサブプログラム毎に記述される宣言である、請求項2に記載のコンパイラである

【0017】

本特許出願に係る請求項4に記載の発明は、原始プログラムを目的プログラムに変換するコンパイラであって、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第5のサブプログラムから呼び出される第6のサブプログラムを含む他プログラムモジュールに含まれる、呼び出された後のステップを、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、を含む前記目的プログラムのステップに変換する、コンパイラである。

【0018】

本特許出願に係る請求項5に記載の発明は、原始プログラムを目的プログラムに変換するコンパイラであって、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第7のサブプログラムから呼び出される第8のサブプログラムを含む他プログラムモジュールの記述に基づいて、前記原始プログラムに含まれる他プログラムモジュールに含まれる、呼び出された後のステップを、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用データメモリ領域アドレスを読み出して設定するステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、を含む前記目的プログラムのステップに変換し、或いはまた、前記原始プログラムに含まれる他プログラムモジュールに含まれる前記他プログラムモジュール用データメモリ領域をアクセスす

るステップを、前記呼び出し元プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイラである。

【0019】

本特許出願に係る請求項6に記載の発明は、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第7のサブプログラムから呼び出される第8のサブプログラムを含む他のプログラムモジュールの記述は、前記第8のサブプログラム毎に記述される宣言である、請求項5に記載のコンパイラである。

【0020】

本特許出願に係る請求項7に記載の発明は、前記呼び出される第2のサブプログラムまたは第4のサブプログラムを含む他プログラムモジュールを含む前記原始プログラムは、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共用データメモリ領域を使用するのか、または、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの実行毎に固有にアクセスする前記他プログラムモジュール用の実行固有データメモリ領域を使用するのか、に関する記述を含む、請求項1から請求項3のいずれか1項に記載のコンパイラである。

【0021】

本特許出願に係る請求項8に記載の発明は、前記他プログラムモジュール用の実行共有データメモリ領域を使用するのかまたは前期他プログラムモジュール用の実行固有データメモリ領域を使用するのかに関する記述は、前記他のプログラムモジュールに含まれる外部変数毎に記述される宣言である、請求項7に記載のコンパイラである。

【0022】

本特許出願に係る請求項9に記載の発明は、前記原始プログラムに含まれる第1のサブプログラムまたは第3のサブプログラムから第2のサブプログラムまたは第4のサブプログラムを呼び出すステップを、前記呼び出し元プログラムモジ

ジュール用の実行共有データメモリ領域アドレスを保存するステップと、前記呼び出される他のプログラムモジュール用の実行共有データメモリ領域アドレスを設定するステップと、前記呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用の実行固有データメモリ領域アドレスを設定するステップと、前記第1のサブプログラムまたは第3のサブプログラムから前記第2のサブプログラムまたは第4のサブプログラムへの移行ステップと、前記第2のサブプログラムまたは第4のサブプログラムから前記第1のサブプログラムまたは第3のサブプログラムへの復帰後に、前記保存した呼び出し元プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して再設定するステップと、前記保存した呼び出し元プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して再設定するステップと、を含む目的プログラムのステップに変換することがあり、前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行固有データメモリ領域をアクセスするステップを、前記設定された他プログラムモジュール用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行共有データメモリ領域をアクセスするステップを、前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、を含む目的プログラムのステップに変換する、請求項7または請求項8のいずれかに記載のコンパイラである。

【0023】

本特許出願に係る請求項10に記載の発明は、前記呼び出される第6のサブプログラムまたは第8のサブプログラムを含む他プログラムモジュールを含む前記原始プログラムは、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域を使用するのか、または、前記他プログラムモジュールに含まれる外部変数が前記呼び出し元プログラムモジュール

の実行毎に固有にアクセスする前記他プログラムモジュール用の実行固有データメモリ領域を使用するのか、に関する記述を含む、請求項4から請求項6のいずれか1項に記載のコンパイラである。

【0024】

本特許出願に係る請求項11に記載の発明は、前記他プログラムモジュール用の実行共有データメモリ領域を使用するのかまたは前記他プログラムモジュール用の実行固有データメモリ領域を使用するのかに関する記述は、前記他のプログラムモジュールに含まれる外部変数毎に記述される宣言である、請求項10に記載のコンパイラである。

【0025】

本特許出願に係る請求項12に記載の発明は、前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、呼び出された後のステップを、前記呼び出し元プログラムモジュールの実行単位に記憶している各プログラムモジュール用データメモリ領域アドレステーブルから前記他プログラムモジュール用の実行固有データメモリ領域アドレスを読み出して設定するステップと、前記呼び出し元プログラムモジュールの複数の実行が共通にアクセスする前記他プログラムモジュール用の実行共有データメモリ領域アドレスを読み出して設定するステップに変換し、前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行固有データメモリ領域をアクセスするステップを、前記設定された他プログラム用の実行固有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、前記原始プログラムに含まれる前記他プログラムモジュールに含まれる、前記他プログラムモジュール用の実行共有データメモリ領域をアクセスするステップを、前記設定された他プログラムモジュール用の実行共有データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップと、を含む前記目的のプログラムのステップに変換する、請求項10または請求項11に記載のコンパイラである。

【0026】

本特許出願に係る請求項13に記載の発明は、前記呼び出し元プログラムモジ

ジュール用の実行共有、および、実行固有データメモリ領域アドレスを保存するステップは、前記第1のサブプログラムまたは第3のサブプログラムから第2のサブプログラムまたは第4のサブプログラムへ移行するために必要なデータよりも先に前記呼び出し元プログラムモジュール用の実行共有、および、実行固有データメモリ領域アドレスをスタックメモリに保存するステップを含む前記目的プログラムのステップである、請求項1から請求項3または請求項7から請求項9のいずれか1項に記載のコンパイラである。

【0027】

本特許出願に係る請求項14に記載の発明は、前記原始プログラムに含まれる前記呼び出し元プログラムモジュールと、前記呼び出し元プログラムモジュールに含まれる第1のサブプログラムまたは第3のサブプログラムまたは第5のサブプログラムまたは第7のサブプログラムから呼び出される第2のサブプログラムまたは第4のサブプログラムまたは第6のサブプログラムまたは第8のサブプログラムを含む他プログラムモジュールとは、同じプログラムモジュールである、請求項1から請求項13のいずれか1項に記載のコンパイラである。

【0028】

本特許出願に係る請求項15に記載の発明は、コンピュータ読み取り可能な記録媒体であって、請求項1から請求項14のいずれか1項に記載のコンパイラを記録した記録媒体である。

【0029】

本特許出願に係る請求項16に記載の発明は、請求項1から請求項14のいずれか1項に記載のコンパイラによって生成される目的プログラムである。

【0030】

本特許出願に係る請求項17に記載の発明は、コンピュータ読み取り可能な記録媒体であって、請求項16に記載の目的プログラムを記録した記録媒体である。

【0031】

本特許出願に係る請求項18に記載の発明は、原始プログラムを目的プログラムに変換するコンパイル方法であって、前記原始プログラムに含まれる呼び出し

元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムを呼び出すステップを、前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第1のサブプログラムから前記第2のサブプログラムへの移行ステップと、前記第2のサブプログラムから第1のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイル方法である。

【0032】

本特許出願に係る請求項19に記載の発明は、原始プログラムを目的プログラムに変換するコンパイル装置であって、前記原始プログラムに含まれる呼び出し元プログラムモジュールに含まれる第1のサブプログラムから他プログラムモジュールに含まれる第2のサブプログラムを呼び出すステップを、前記呼び出し元プログラムモジュール用データメモリ領域アドレスを保存するステップと、前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップと、前記第1のサブプログラムから前記第2のサブプログラムへの移行ステップと、前記第2のサブプログラムから第1のサブプログラムへの復帰後に前記保存した呼び出し元プログラムモジュール用データメモリ領域アドレスを読み出して再設定するステップと、を含む前記目的プログラムのステップに変換し、前記原始プログラムに含まれる他プログラムモジュールに含まれる、前記他プログラムモジュール用データメモリ領域をアクセスするステップを、前記設定された呼び出される他プログラムモジュール用データメモリ領域アドレスからの相対アドレスで、データメモリ領域アクセスするステップを含む前記目的プログラムのステップに変換する、コンパイル装置である。

【0033】

【発明の実施の形態】

本特許出願に係る発明（以後、「本発明」とも言う）の実施の形態について図を参照して説明する。

【0034】

（実施の形態1）

図1は本発明の第1の実施の形態であるコンパイラが変換するソースプログラムとそのコンパイラによって変換されたオブジェクトプログラムの構造を示す図である。

【0035】

但し、これらのプログラムは必ずしも1つにまとまった形で存在するとは限らず、幾つかの物理的な媒体、ファイル等に分散して記録されていることもある。

【0036】

これらのプログラムは、アプリケーションプログラムモジュール121とアプリケーションプログラムモジュール131とライブラリプログラムモジュール151と、その他本発明の説明に直接関係しないので図示していない幾つかのプログラムモジュールとを含んでいる。

【0037】

プログラムモジュールとは、プログラムを何らかの単位（例えば、機能単位、リンク単位、バージョンアップ単位等）で分割したものであり、プログラムコード領域と固有データ領域を保持する。尚ここでは固有のデータ領域としたが、必ずしも厳密に固有であることは必要でなく、他の何らかの目的や用途と兼用しているものであっても良い。このことは他の説明についても同様である。

【0038】

プログラムモジュールのプログラムコード領域とは、そのプログラムモジュール内に記述されたプログラムの実行命令等を保持する領域である。

【0039】

プログラムモジュールの固有データ領域とは、そのプログラムモジュール内で定義された、外部変数、および静的変数の実体を保持する領域である。

【0040】

以後、外部変数と静的変数の両方を含めて、外部変数と言うこともある。

【0041】

アプリケーションプログラムモジュール121は、プログラムコード領域123とこのアプリケーションプログラムモジュール121の固有データ領域125と、その他本発明の説明に直接関係しないので図示していない幾つかの部分とを含んでいる。

【0042】

アプリケーションプログラムモジュール131は、プログラムコード領域133とこのアプリケーションプログラムモジュール131の固有データ領域135と、その他本発明の説明に直接関係しないので図示していない幾つかの部分とを含んでいる。

【0043】

これらのプログラムはアプリケーションプログラムモジュール121とアプリケーションプログラムモジュール131は、呼び出し元プログラムであって、共通にライブラリプログラムモジュール151を呼び出す。この呼び出しは通常、関数コール、サブルーチンコール等の形態で行われる。

【0044】

それが図1に示すプログラム呼び出し1とプログラム呼び出し2（再入：reentry）である。

【0045】

これらプログラムモジュールは通常、幾つかのサブプログラムを含んでいる。このサブプログラムは通常、関数やサブルーチンやその他である。また、これら関数やサブルーチン1つをプログラムモジュールと呼ぶこともある。

【0046】

また通常、プログラムモジュールの呼び出しは、他プログラムモジュールのサブプログラムを呼び出すことによって行うが、同一プログラムモジュール内のサブプログラムを呼び出すこともある。

【0047】

いわゆるマルチタスク環境、その他時分割的に複数のプロセスまたはタスクまたはアプリケーションが並行して実行される、或いは実行しなければならない環境では、例えば、アプリケーションプログラムモジュール121がライブラリプログラムモジュール151を呼び出して実行中に、並行して、アプリケーションプログラムモジュール131がライブラリプログラムモジュール151を呼び出して実行することが発生する。

【0048】

このような状態は種々の情報機器で無数に近く発生するが、非常に簡単な幾つかの例を示す。

【0049】

例えば、携帯電話等のモバイル情報通信機能を有する機器では、ユーザーが送信するためのデータを入力している時にも、同時に並行して着信の有無を監視していなければならない。

【0050】

このデータ入力と着信の監視を別のCPUで行うこともできるが、CPUの節約のため1CPUで行おうとするならば、時分割的に並行して行う必要が起こる。

【0051】

そしてある基本的な関数型サブルーチンが、これらデータ入力アプリケーションと着信監視アプリケーションとの両方で使用される関数型サブルーチンであったならば、この関数型サブルーチンは、一方のアプリケーションから呼び出されて実行中に、もう一方のアプリケーションから呼び出されることが起こり得る。

【0052】

このように複数のタスク等が時間的に重複して同一のプログラムモジュール（関数またはサブプログラムまたはサブルーチン等）を呼び出し、実行するようなケースを「再入：reentry」と言い、あるプログラムモジュールが時間的に重複・並行して複数のタスクを実行中のプログラムから呼び出され、実行され得るように構成されていることを「再入可能：reentrant」と言う。

【0053】

従来から、プログラムモジュールを再入可能であるように構成する方法は幾つか知られている。

【0054】

その第1は、いわゆる外部変数を使用せず、全て内部変数（以後、Local変数と言うこともあるが同一内容を表す）で処理することであった。内部変数はこのプログラムモジュールが呼び出される毎にスタック領域に新たに確保され、このプログラムモジュールの処理が終了するのと同時に解放されるため、全て内部変数を使用している限り、変数へのアクセスが衝突することは発生せず、同時に並行的に実行されても問題はなかった。

【0055】

その第2は、いわゆる外部変数を使用するが、この外部変数に対するアクセスをいわゆるセマフォ（semaphore：腕木信号機）と呼ばれる状態変数によって調停する方法である。

【0056】

第1の方法では、外部変数を使用することができないため、このプログラム以外のプログラムモジュールとデータを受け渡しするのに制約を受けることがある。

【0057】

第2の方法では、煩雑なセマフォに対するテスト・アンド・セットを外部変数にアクセスする毎に行わなければならない、多くの負担をプログラマに課し、間違いや誤動作の原因となることが多かった。

【0058】

ライブラリプログラムモジュール151は、プログラムコード領域153とこのライブラリプログラムモジュール151の固有データ領域155、157と、その他本発明の説明に直接関係しないので図示していない幾つかの部分とを含んでいる。

【0059】

アプリケーションプログラムモジュール121用データ領域155とアプリケーションプログラムモジュール131用データ領域157とは、このライブラリ

プログラムモジュール 151 を呼び出した元のプログラムモジュールによって区別されている。

【0060】

例えば、アプリケーションプログラムモジュール 121 からこのライブラリプログラムモジュール 151 が呼び出された時には、プログラムコード領域 153 は外部変数領域としてアプリケーションプログラムモジュール 121 用データ領域 155 をアクセスしてその処理を行い、アプリケーションプログラムモジュール 131 からこのライブラリプログラムモジュール 151 が呼び出された時には、プログラムコード領域 153 は外部変数領域としてアプリケーションプログラムモジュール 131 用データ領域 157 をアクセスしてその処理を行う。

【0061】

このように呼び出し元プログラム単位に、ライブラリプログラムモジュールが使用する固有データ領域、特に外部変数のために使用するデータ領域を区分して使用することによって、外部変数に対するアクセスが衝突することを気にせず、プログラムを作成することができる。

【0062】

既に説明したように、内部変数については、このライブラリプログラムモジュール 151 が呼び出される毎に、そのために領域がスタックに確保されるため、アクセスが衝突する問題は生じない。この状態の例を図 4 (b) の Local 変数 1 ~ 3 に示す。

【0063】

尚ここで、データ領域を呼び出し元アプリケーションプログラム単位に用意して使用すると説明したが、ここで言うアプリケーションプログラムを、アプリケーションタスク単位と読み替えても良く、ここで言うアプリケーションプログラムはアプリケーションタスクと同じ概念であると考えても良い。

【0064】

また、ここで言うアプリケーションプログラムを、アプリケーションプロセス単位と読み替えても良く、ここで言うアプリケーションプログラムはアプリケーションプロセスと同じ概念であると考えても良い。

【0065】

より詳しくアプリケーションプログラムモジュール121がライブラリプログラムモジュール151を呼び出す時の処理手順を図2と図3に示す。

【0066】

図3(a)、(b)に示すソースプログラムは、C言語を例として記述している。

【0067】

アプリケーションプログラムモジュール121のソースプログラムに記述された「bar(A, B);」命令がオブジェクトプログラムに変換されると、ライブラリプログラムモジュール151に含まれる関数型サブルーチンであるbar(以後、関数型サブプログラムbarと言うこともあり、単に関数barと言うこともあるが同一である)を呼び出すオブジェクトプログラムが生成される(図3(a)の1~7)。

【0068】

アプリケーションプログラムモジュール121のプログラムコード領域123に含まれるプログラムは、ライブラリプログラムモジュール151に含まれる関数barを呼び出すに際し、まず現在実行中のアプリケーションプログラムモジュール121が使用しているデータ領域125の先頭アドレスである0x0600(16進数表記で0600であり、10進数表記に変換すると、 $0 \times 16^3 + 6 \times 16^2 + 0 \times 16 + 0$ である、以後同じ)を保存しているDPレジスタ(以後単にDPと言うこともある)の内容をスタックに保存する(図2(a)の1、図3(a)の1))。

【0069】

これはライブラリプログラムモジュール151から戻ってきた時に、アプリケーションプログラムモジュール121がDPの内容を再度0x0600に設定し、アプリケーションプログラムモジュール121が自分自身のための固有データ領域125を再度使用することができるようにするためである。尚ここでは固有のデータ領域としたが、必ずしも厳密に固有であることは必ずしも必要でなく、他の何らかの用途や目的、その他と共有であっても良い。このことは他の説明に

についても同様である。

【0070】

DPレジスタは、実行中のプログラムモジュールのデータ領域の先頭アドレスを保持する領域である。この領域として、通常特別なレジスタを用意するが、必ずしも特別なレジスタを用意せず、汎用的なレジスタを使用してもよい。

【0071】

次に、アプリケーションプログラムモジュール121に含まれるプログラムは、ライブラリプログラムモジュール151に引き渡すデータである引数Aと引数Bをスタックに保存する（図2（a）の2、図3（a）の2）。

【0072】

この時のスタックの状態を図4（a）に示す。

【0073】

図4（a）に示すようにスタック（以後、スタックメモリと言うこともある）はLast-in/First-out型のメモリ領域で、あるプログラムモジュールから他のプログラムモジュール（例えば、関数やサブルーチン、サブプログラム、その他である）を呼び出す時に必要なデータを保存して、プログラムモジュール間でデータの引き渡しに使ったり、Local変数用メモリ領域として使用したり、呼び出された関数等から呼び出し元のプログラムに戻るためのアドレスの保存や、その他の目的に使用されるものである。

【0074】

スタックのデータを出し入れする領域のアドレスは、スタックポインタと呼ばれるレジスタで示される（以後、単にスタックポインタ或いはスタックポインタレジスタと呼ぶこともある）。

【0075】

このスタックの使われ方を例えば図4（b）に示す。例えば図4（b）に示すスタックの上から3番目に格納されているローカル変数2に対してアクセスするには、スタックポインタの値を変更するか或いは、スタックポインタレジスタの値マイナス2の値をアドレスとして間接参照を行うこともできる。

【0076】

次に、アプリケーションプログラムモジュール121に含まれるプログラムは、自分自身の固有データ領域125内の領域に記憶している、関数b a rを含むライブラリプログラムモジュール151で使用する、アプリケーションプログラムモジュール121固有のデータ領域155のアドレス(0x800)を読み出し、そのアドレスをDPに設定する(図2(a)の3、図3(a)の3)。

【0077】

前記関数b a rを含むライブラリプログラムモジュール151で使用する、アプリケーションプログラムモジュール121固有のデータ領域155のアドレス(0x800)を保持する領域は、通常コンパイラが設定する。しかし必ずしも、前記領域をコンパイラが設定せず、ライブラリとして用意し、リンク時に設定してもよい。

【0078】

次に、アプリケーションプログラムモジュール121に含まれるプログラムは、呼び出し先プログラムである関数b a rから自分自身に帰ってくる時のアドレス、即ち関数b a rへのジャンプ命令の次の命令のアドレスをスタックに保存し、関数b a rへジャンプする(図2(a)の4、図3(a)の4)。

【0079】

図2(b)に示すように、呼び出されるプログラムである関数b a rは、ライブラリプログラムモジュール151に含まれるプログラムコード領域153に含まれている。

【0080】

この関数b a rは引数としてスタックに保存された引数Aと引数Bとを取り出して処理に使用する。この時のスタックの状態は前記図4(a)に示す通りである。

【0081】

更にこの関数b a rは外部変数として、例えばこのライブラリプログラムモジュール151で定義された外部変数Cを使用する(図2(b)の5、図3(b)の5)。

【0082】

関数 `bar` ではこの外部変数 `C` に対してアクセスし、例えばそのメモリ領域に値 `10` を代入するに際し、`DP` に保存されている値 (`0x800`) に `DP` からの相対アドレス (`0x50`) を加算したアドレス (`0x850`) に値 `10` を代入する。この命令形式を図 2 (b) の 5 と図 3 (b) の 5 に示す。

【0083】

これによって関数 `bar` は、アドレス `0x800` から始まるメモリ領域であるアプリケーションプログラムモジュール 121 用の固有データ領域 155 に対してアクセスすることができる (図 2 (b) の 5)。

【0084】

関数 `bar` での処理が終ると、プログラムの制御は元のアプリケーションプログラムモジュール 121 に含まれるプログラムに戻される。

【0085】

関数 `bar` からの復帰直後に、スタックから引数 `A`、`B` が破棄される (図 2 (a) の 6、図 3 (a) の 6)。尚ここでは関数 `bar` からの復帰直後としたが、必ずしも厳密に直後であることは必要でなく、何らかの処理の後であっても良い。このことは他の説明についても同様である。

【0086】

次に、スタックに保存していた元の `DP` の値、この例では `0x600` がスタックから取り出されて `DP` に戻される (図 2 (a) の 7、図 3 (a) の 7)。

【0087】

これによって呼び出し元のアプリケーションプログラムモジュール 121 に含まれるプログラムは、再度自分自身の固有データ領域 125 に対してアクセスすることが可能になる (図 2 (a))。

【0088】

尚この時、図 4 に示す通り、呼び出し元の `DP` の値は、スタックの一番下に保存されているので、呼び出されたライブラリプログラムモジュール 151 の側ではその内容を読み出す必要がないと共に、実際に読み出さなくても他の必要な処理を全て行うことが可能であり、不要なアクセスを行う必要が生じない。

【0089】

また例えば次の第2の実施の形態に示すような他の実施の形態であって、DPをスタックに保存するか保存しないか選択ができる実施の形態では、DPをスタックの一番下に保存することによって、DPがスタックに保存されているか保存されていないかに係わらず、同一アドレスに対するアクセスによって同じ処理を行うことができ、DPがスタックに保存されているか否かを判断して処理を分ける必要がない。

【0090】

即ち、DPをスタックの一番下に保存することによって、DPがスタックに保存されているか保存されていないかに係わらず、呼び出されたライブラリプログラムモジュール151の側に必要なデータ、例えば引数や帰りアドレスを取り出すスタック位置は変わらない。

【0091】

即ち、引数や帰りアドレスを取り出すスタックのスタックポイントからの相対位置は変化しないので、それを考慮してこれら引数や帰りアドレスを取り出すことは必要でない。

【0092】

尚、この例では、アプリケーションプログラムモジュールから、ライブラリプログラムモジュールを呼び出す例について書いたが、ライブラリプログラムモジュールから、他のライブラリモジュールを呼び出すことも可能であり、その際の処理は、アプリケーションプログラムモジュールから、他ライブラリモジュールを呼び出す場合の処理と同じであるから、説明を省略する。

【0093】

尚、この例では、他のプログラムモジュールを呼び出す例について書いたが、同一プログラムモジュールを呼び出すことも可能であり、その際の処理は、他のプログラムモジュールを呼び出す場合の処理と同じであるから、説明を省略する。

【0094】

(実施の形態2)

前記本発明の第1の実施の形態では、アプリケーションプログラムモジュール

121のソースプログラムに記述された「bar (A, B);」命令がオブジェクトプログラムに変換されることによって生成されるオブジェクトプログラムは、現在実行中のアプリケーションプログラムモジュール121が使用しているデータ領域125の先頭アドレスを保存しているDPレジスタの内容をスタックに保存し、ライブラリプログラムモジュール151に引き渡すデータである引数Aと引数Bをスタックに保存し、呼び出される関数barで使用する、アプリケーションプログラムモジュール121固有のデータ領域155のアドレスを読み出してそのアドレスをDPに設定し、関数barから自分自身に帰ってくる時のアドレスをスタックに保存し、関数barへジャンプし、関数barからの復帰時には、引数Aと引数Bをスタックから破棄し、スタックに保存している元のDPの値を再設定するステップを含むものであった。

【0095】

次に説明する本発明の第2の実施の形態では、上記のような処理を行うか、それとも上記の処理の一部を行わずに省略するかの記述を、ライブラリプログラムモジュール151のソースプログラムの記述は含むものである。そのための記述を図8に示す。

【0096】

本発明の第1の実施の形態では、他プログラムモジュールの呼び出しを例に挙げ説明したが、本発明の第2の実施の形態では、同一モジュール内の関数barの呼び出しを例に挙げて説明する。

【0097】

図8ではC言語を例に記述している。この例ではC言語の言語仕様を拡張して、新規キーワードとして「private」を追加し、ソースコード上で「private」修飾詞をつけることにより、宣言を記述している。

【0098】

例えば図8に示す関数barの呼び出しでは、上記本発明の第1の実施の形態で行った処理の一部を行うことなく、省略できることを示す。

【0099】

そのための宣言が、例えば図8に示す関数barでは、「private v

oid bar (int A, int B)」である。

【0100】

図8に示すように、ライブラリモジュールプログラムのソースプログラムに含まれる関数barには、関数barが「private」である旨の記述が含まれている。

【0101】

この記述は「関数barがいわゆるプライベート (private) 関数である」旨を宣言する記述であり、例えば「プライベート (private) 宣言」或いは「プライベート (private) 関数宣言」と呼ばれることもある。

【0102】

このプライベート関数barを呼び出す記述が行われると、第1の実施の形態で説明したような、呼び出し元であるプログラムモジュールが使用しているDPの保存や、呼び出されるプログラムモジュールで使用される固有データ領域アドレスを示すDPの設定や、関数barからの復帰時に、スタックに保存している元のDPの値の再設定を行うためのオブジェクトプログラムのステップは生成されない。

【0103】

このプライベート関数barを呼び出す記述がライブラリプログラムモジュール151のプログラムコード領域153で行われると、図7に示すように、関数barで使用するためのデータである引数Aや引数Bを関数barに引き渡すためにスタックへ保存するステップと、関数barへジャンプする命令の次の命令のアドレスをスタックへ格納し、関数barへジャンプするステップと、関数barからの復帰時に、引数Aや引数Bをスタックから破棄するステップと、を含むオブジェクトプログラムが本実施の形態のコンパイラによって生成される。

【0104】

これによって、必ずしも呼び出し元であるプログラムモジュールと呼び出されるプログラムモジュールとが異なるデータ領域を使用することが必要でないケースでは、関数barをプライベート宣言することによって、DPの保存や設定や再設定のステップを省略し、オブジェクトプログラムの実行を高速化することが

でき、不要なステップを出力しないことでオブジェクトプログラムサイズを小さくすることもできる。

【0105】

このようなプライベート宣言された関数 `bar` が実行されると、図7に示すように、DPの内容は呼び出し元プログラムであるプログラムモジュールの状態のまま変化していないので、例えば外部変数Cに対してデータの読み出し等が実行されると、呼び出し元であるプログラムモジュールの固有データ領域（この場合、固有データ領域155）がアクセスされる。

【0106】

また、図示していないが、プライベート宣言された関数 `bar` から呼び出し元プログラムへリターンした時にも、DPの再設定等を行う必要はない。

【0107】

またこれまで説明した実施の形態では、例えば関数 `bar` の前に「`private`」と記述されるとプライベート関数であることが宣言され、DPの保存等の一連の処理が行われず、特に何も記述されなければプライベート関数でなく、DPの保存等の処理が行われる。

【0108】

しかし他の実施の形態では、例えば明示的にプライベート関数でないことを示すために「`public`」等の記述を行うことも可能である。

【0109】

このような記述は、「パブリック（`public`）宣言」或いは「パブリック（`public`）関数宣言」と呼ばれることもあり、この記述を行うことによって明示的にDPの保存等の一連の処理を行うことを示すことができる。

【0110】

また他の実施の形態では、例えばこの「プライベート宣言」が行われる関数は、同一プログラムモジュール内でのみ呼び出される、或いは呼び出すことが可能な関数であっても良く、更に他の実施の形態では、この「プライベート宣言」が行われる関数は、異なるプログラムモジュール間でのみ呼び出される、或いは呼び出すことが可能な関数であっても良い。

【0111】

尚、この例では新規キーワード「private」や「public」を追加し、C言語を拡張して宣言を行っているが、プラグマ等を用いることにより、言語使用を拡張することなく、宣言を行ってもよい。

【0112】

また、別ファイルとして記述したり、コンパイラの引数として指定してもよい。

【0113】

(実施の形態3)

前記本発明の第1の実施の形態では、関数barが使用する全ての外部変数（例えばC）に対してアクセスするに際し、DPに保存されている値（例えば、0x800）に、DPからの相対アドレス（例えば、0x50）を加算したアドレス（例えば、0x850）でアクセスした。

【0114】

その結果、関数barは、使用する全ての外部変数用領域として、アドレス例えば、0x800から始まるメモリ領域であるアプリケーションプログラムモジュール121用固有データ領域155を使用することができる。

【0115】

しかし次に説明する本発明の第3の実施の形態では、例えば関数barを呼び出す複数のアプリケーションプログラムモジュールが共通にアクセスするデータメモリ領域に関する記述を、例えば関数barの原始プログラムはその変数宣言部に含むことができる。

【0116】

その記述の例を図6示す。

【0117】

図6では、C言語を例として宣言を記述している。この例ではC言語の言語仕様を拡張して新規のキーワード「common」を追加し、ソースコード上でこの「common」修飾詞をつけることにより宣言を記述している。

【0118】

例えば図6に示す関数 `bar` のプログラムでは、関数 `bar` が使用する外部変数として `D` と `C` を使用する (図6 (a))。

【0119】

その宣言が例えば図6の「`common int D;`」と「`int C;`」の記述である (図6 (a))。

【0120】

そして更に「`common int D;`」の記述は、この外部変数 `D` はこの関数 `bar` の複数の呼び出し元アプリケーションプログラムが共通にアクセスする外部変数であることを宣言している (図6 (a))。

【0121】

尚、この例では新規のキーワード「`common`」を追加し、`C` 言語を拡張して宣言を行っているが、プラグマ等を用いて言語仕様を拡張することなく宣言を行ってもよい。

【0122】

また、別ファイル等に記述したり、コンパイラの引数として指定することによって宣言を行ってもよい。

【0123】

図6に示すように、ライブラリモジュールプログラムのソースプログラムに含まれる外部変数 `D` には、この外部変数 `D` が、「`common`」である旨の記述が含まれている。

【0124】

この記述は、「外部変数 `D` がいわゆる共通外部変数である」旨を宣言する記述であり、例えば「`コモン (common) 宣言`」或いは「`コモン (common) 外部変数宣言`」と呼ばれることもある。

【0125】

図5 (a) は、プログラムコード領域553が、再入可能な関数 `bar` に相当し、このプログラムから固有データ領域であるアプリケーションプログラムモジュール121用データ領域555と、共有データ領域である共有データ領域559との両方がアクセスされ得ることを示している (図5 (a))。尚ここでは共

有データ領域としたが、必ずしも厳密に他の何らかの目的や用途と共有されていることは必要でなく、たまたま他の何らかの目的や用途と共有されていないものであってもかまわない。このことは他の説明についても同様である。

【0126】

図5 (b) はより具体的に、外部変数Cは固有データ領域であるアプリケーションプログラムモジュール121用データ領域555に格納され、外部変数Dは共有データ領域アクセスによってアクセスされる共有データ領域559に格納されることを示している (図5 (b))。

【0127】

図6の「int C;」の記述は、この外部変数Cは、この関数barの複数の呼び出し元アプリケーションプログラム毎に固有のメモリ領域を使用することを宣言している記述であり、この記述によって、外部変数Cは固有データ領域アクセスによってアクセスされるアプリケーションプログラムモジュール (この例ではプログラムモジュール121) 用の固有のデータ領域555に格納される (図5 (b))。

【0128】

そして関数barでは、呼び出し元アプリケーションプログラム (例えばプログラムモジュール121) 毎に固有のデータ領域555に格納されている外部変数Cに対してアクセスし、例えばそのメモリ領域に、値10を代入するに際し、DPに保存されている値 (例えば、0x800) に、DPからの相対アドレス (例えば、0x50) を加算したアドレス (例えば、0x850) に、値10を代入する。この命令形式を図5 (b) と図6に示す。

【0129】

この時の命令形式は前記第1の実施の形態と同一である。

【0130】

これによって関数barは、例えばアドレス0x800から始まるメモリ領域であるアプリケーションプログラムモジュール121用の固有データ領域555に対してアクセスすることができる (図5 (b))。これによって関数barに複数のアプリケーションプログラムモジュール (例えば、図1 (a) のアプリケ

ーションプログラムモジュール121とアプリケーションプログラムモジュール131)から同時に呼び出され、その結果再入が起こっても外部変数へのアクセスが衝突することはない。

【0131】

図6の「common int D;」の記述は、この外部変数Dは、関数barの複数の呼び出し元アプリケーションプログラムが共有するメモリ領域を使用することを宣言している記述であり、この記述によって、外部変数Dは共有データ領域アクセスによってアクセスされる複数のアプリケーションプログラムモジュールが共通に使用する共有データ領域559に格納される(図5(b))。

【0132】

そして関数barでは、呼び出し元アプリケーションプログラム(例えばプログラムモジュール121)が共通に使用する共有メモリ領域559に格納されている外部変数Dに対してアクセスし、例えばそのメモリ領域に、値20を代入するに際し、特別なレジスタSDPに保存されている値(例えば、0xF00)に、SDPからの相対アドレス(例えば、0x60)を加算したアドレス(例えば、0xF60)に、値20を代入する。この命令形式を図5(b)、図6に示す。

【0133】

このSDPは特別なレジスタであり、全アプリケーションプログラムに対して1つの共有データ領域の先頭アドレスを保存している。またこのSDPに保存されている値である全アプリケーションプログラムに対して1つの共有データ領域の先頭アドレス自体をSDPと呼ぶこともある。

【0134】

或いは他の実施の形態ではこのSDPとして必ずしも特別なレジスタを使用せず、汎用的なレジスタを使用することもできる。このような実施の形態は汎用のレジスタを多く備えているようなMPUにおいて特に有効であり、更にこの汎用レジスタがメモリアクセスのためのオフセットアドレスとして使用できるMPUでは更に有効である。

【0135】

或いはまた他の実施の形態ではこのSDPとして必ずしも特別なレジスタや汎用のレジスタを使用せず、通常のメモリやスタックを使用することもできる。

【0136】

しかしこのような実施の形態ではこのスタックや通常のメモリに格納されたSDPをオフセットアドレスとして使用してメモリアクセスするために、幾らかのステップが必要になることもある。

【0137】

例えばそのステップとは、スタックやメモリに格納されているSDPを読み出し、メモリアクセスのためのオフセットアドレスとして使用できるレジスタに格納したり、そのための準備として現在のレジスタの内容を退避するステップ等である。

【0138】

これらのような方法で、このSDPレジスタが記憶している値をオフセットアドレスとして使用してメモリ領域アクセスすることによって、関数barは、例えばアドレス0xF00から始まるメモリ領域であるアプリケーションプログラムモジュール共有データ領域559に対してアクセスすることができる(図5(b))。

【0139】

SDPの値は、図示していないが例えば図2(a)のデータ領域125の、ライブラリプログラムモジュール151の自分固有データ領域アドレスと同様にデータ領域125に記憶しておき、アプリケーションプログラムモジュール121からライブラリプログラムモジュール151を呼び出す時、その呼び出しステップの中で保存と設定を行うこともできる。

【0140】

例えば、図2(a)で「1. 現在のDP(0x0600)をスタックに保存」するステップの前のステップとして、「(0.5) 現在のSDPをスタックに保存」するステップを含めることも可能である。

【0141】

また図2(a)で「3. 新しいDP(0x0880)を設定」するステップの

前のステップとして、「(2.5) 新しいSDP (0x0F00) を設定」するステップを含めることも可能である。

【0142】

このように現在のSDPをスタックに保存するステップを、引数Aや引数Bをスタックに保存するステップよりも先に行うことによって、呼び出されるライブラリプログラムモジュールの側でこのスタックに保存されたSDPの有無に関与しなくてもよいことは、前記第1の実施の形態と同じであるから詳細な説明を省略する。

【0143】

このような実施の形態によって、関数barが、複数のアプリケーションプログラムモジュール（例えば、図1(a)のアプリケーションプログラムモジュール121とアプリケーションプログラムモジュール131）から同時に呼び出され、その結果再入が起こった時、明示的に、その外部変数の幾つかを共通に使用し、データの共有、受け渡し、その他の処理を行うことができる。

【0144】

またこの時、前記したセマフォ等を使った調停が行われることがある、或いは行わなければならないことも発生し得るが、本発明の内容と直接関係しないので説明を省略する。

【0145】

(実施の形態4)

図9に本発明の第4の実施の形態であるコンパイラが変換の対象とする呼び出し元プログラムであるアプリケーションプログラムモジュール121と、呼び出されるプログラムであるライブラリプログラムモジュール151と、その呼び出し処理時に行われる処理の一例を示す(図9(b)、図9(c))。

【0146】

この実施の形態では、呼び出し元アプリケーションプログラムモジュール単位、或いは呼び出し元アプリケーションプログラム単位、或いは呼び出し元アプリケーションプロセス単位、或いは呼び出し元アプリケーションタスク単位、或いは呼び出し元アプリケーション単位、或いは呼び出し元のプロセス単位に、或い

はまた呼び出し元のタスク単位に、CCT (Context Control Table) と呼ばれる、各プログラムモジュールの固有のデータ領域へのアドレスを保存しているテーブルを備えている (図9 (a))。

【0147】

呼び出し元のアプリケーションプログラムモジュール121では、アプリケーションプログラム121用CCT (図9 (a)) を参照してアプリケーションプログラムモジュール121自体のデータ領域アドレスであるこの例では0x600を取り出し、この値をDPに設定し、この値をオフセットアドレスとして使用して外部変数アクセスすることによって、アプリケーションプログラムモジュール121用の固有データ領域125を使用する。

【0148】

このアプリケーションプログラムモジュール121からライブラリプログラムモジュール151に含まれる関数barを呼び出すには、まず関数barに引き渡す引数Aと引数Bをスタックに保存する。

【0149】

次に、アプリケーションプログラムモジュール121ではこの関数barから自分自身へ帰ってくるためのアドレス、即ち関数barへのジャンプ命令の次のアドレスをスタックに保存し、関数barに制御を移行する。

【0150】

制御の移行を受けた呼び出し先ライブラリプログラムモジュール151に含まれる関数barでは、現行のDPの内容を保存する。現行のDPの内容とは呼び出し元のアプリケーションプログラムモジュール121用の固有データ領域アドレスの値であり、関数barから呼び出し元のプログラムに戻った時、再度この値をDPに設定してアプリケーションプログラムモジュール121用の固有データ領域を使用するために保存しておくものである。

【0151】

次にこのDPの内容を参照してアプリケーションプログラムモジュール121用CCTへのポインタアドレスを取り出し、更にそのポインタアドレスを参照してアプリケーションプログラムモジュール121用CCTを取り出し、更にその

CCTの内容を参照して、関数 `bar` でアプリケーションプログラムモジュール 121 が固有に使用するアプリケーションプログラムモジュール 121 用データ領域 155 のアドレス（この例では、`0x800`）を取り出し、その内容を DP レジスタに設定する（図 9（c））。

【0152】

これによって、呼び出されたライブラリプログラムモジュール 151 に含まれる関数 `bar` では、外部変数に対してアクセスする時、DP レジスタの値をオフセットアドレスとして使用してアクセスするので、関数 `bar` が使用する全ての外部変数は `0x800` がオフセットアドレスとして使用されてアクセスされ、アプリケーションプログラムモジュール 121 から呼び出される限りアプリケーションプログラムモジュール 121 用データ領域 155 が割り当てられる（図 9（c））。

【0153】

これによって、複数のアプリケーションプログラムモジュールから関数 `bar` に対して再入が発生したとしても、外部変数で衝突が発生することは起こらない。

【0154】

関数 `bar` から呼び出し元のアプリケーションプログラムモジュール 121 に戻る時には、DP の内容を保存しておいた呼び出し元のアプリケーションプログラムモジュール 121 用の固有データ領域アドレスの値に戻してから、リターンする（図 9（c））。

【0155】

これによって、呼び出し元のアプリケーションプログラムモジュール 121 では再度、呼び出し元のアプリケーションプログラムモジュール 121 自体のデータ領域を使用することができる（図 9（b））。

【0156】

この実施の形態では、ライブラリプログラムモジュール 151 の呼び出し側で DP の設定を行わないため、呼び出し側のプログラムの負担を軽くすることができ、呼び出し回数が多い場合等に特に有効である。

【0157】

また図示していないが、図9(c)で関数barが「private」宣言が行われていれば、上記のようなDPの保存やアプリケーションプログラムモジュール121からの呼び出しで使用するbar用のDPを、CCTから取り出して設定等の処理を行わないことは上記第2の実施の形態と同じであり、「public」宣言が行われているならば明示的に、DPの保存やアプリケーションプログラムモジュール121からの呼び出しで使用するbar用のDPをCCTから取り出して設定等の処理等が行われることを示す。これらは前記第2の実施の形態と同じであるから説明は省略する。

【0158】

更にまた図示していないが、このCCTには前記DPだけでなく、前記第3の実施の形態で説明した新しいSDP(0x0F00)を、図2(a)のデータ領域125に格納して呼び出しプログラムモジュール121のプログラムコード領域123でその保存や設定を行うのではなく、例えば図9(a)のCCTと同様のテーブルを用意して、そのテーブルに各プログラムモジュールの共有データ領域の先頭アドレスを格納しておき、呼び出されるライブラリプログラムモジュールである、例えば図9(c)のライブラリプログラムモジュール151のプログラムコード領域153のステップの中で、そのCCTと同様のテーブルとCCTと同様のテーブルに格納されたSDPを読み出して設定することもできる。

【0159】

これについても前記DPの保存と設定と同じであるから詳細な説明は省略する。

【0160】

また、CCTと同様のテーブルは、CCTと異なり、実行環境で1つだけ保持するとしてもよい。

【0161】

また以上説明したような原始プログラム、目的プログラム、その他のプログラムの全てはコンピュータ読み取り可能な記録媒体に格納されて、コンピュータから読み出されて利用され、実行される、ネットワークによって通信される。

【0162】

同時にこのような翻訳を行うコンパイラもまたコンピュータ読み取り可能な記録媒体に格納されて、コンピュータから読み出されてコンパイル（原始プログラムから目的プログラムへの翻訳・変換）が実行され、利用され、ネットワークによって通信される。

【0163】

このコンパイラによってコンピュータが行うのはこのコンパイラによって規定されたコンパイル方法であり、このコンピュータハードウェアとコンパイラとは全体としてコンパイラ装置を構成する。

【0164】

【発明の効果】

以上説明したように、本発明によると、例えば、いわゆる機器組み込みプログラムのようにオブジェクトプログラムを格納することのできるメモリ領域が限られており、しかも近年の携帯情報機器のようにその限られたメモリ領域に高度且つ高機能な情報処理のためのオブジェクトプログラムを格納しなければならないような目的に適う、特にこのような目的に適うオブジェクトプログラムとして再入可能なオブジェクトプログラムを自動生成するためのコンパイラや、当該コンパイラを記録したコンピュータ読み取り可能な記録媒体や、そのための方法や装置を提供することができ、その効果は非常に大きい。

【図面の簡単な説明】

【図1】

本発明の第1の実施の形態であるコンパイラが変換するソースプログラムとそのコンパイラによって変換されたオブジェクトプログラムの構造を示す図

【図2】

本発明の第1の実施の形態のアプリケーションプログラムモジュール121がライブラリプログラムモジュール151を呼び出す時の処理手順を示す図

【図3】

本発明の第1の実施の形態のアプリケーションプログラムモジュール121がライブラリプログラムモジュール151を呼び出す時の処理手順を示す図

【図 4】

本発明の第 1 の実施の形態のスタックの状態を示す図

【図 5】

本発明の第 3 の実施の形態で呼び出されるプログラムモジュールにおける外部変数の宣言に関する記述を示す図

【図 6】

本発明の第 3 の実施の形態で関数を呼び出す複数のプログラムモジュールが共通にアクセスする領域に関する記述を示す図

【図 7】

本発明の第 2 の実施の形態で呼び出されるプログラムモジュールにおける関数の宣言に関する記述を示す図

【図 8】

本発明の第 2 の実施の形態でライブラリプログラムモジュールが処理の一部を行わずに省略するための記述を示す図

【図 9】

本発明の第 4 の実施の形態であるコンパイラが変換の対象とする呼び出し元プログラムと呼び出されるプログラムとその呼び出し時に行われる処理例を示す図

【符号の説明】

- 1 2 1 アプリケーションプログラムモジュール
- 1 2 3 プログラムコード領域
- 1 2 5 データ領域
- 1 3 1 アプリケーションプログラムモジュール
- 1 3 3 プログラムコード領域
- 1 3 5 データ領域
- 1 5 1 ライブラリプログラムモジュール
- 1 5 3 プログラムコード領域
- 1 5 5 アプリケーションプログラムモジュール 1 2 1 用データ領域
- 1 5 7 アプリケーションプログラムモジュール 1 3 1 用データ領域
- 5 5 1 ライブラリプログラムモジュール

553 プログラムコード領域

555 アプリケーションプログラムモジュール121用データ領域

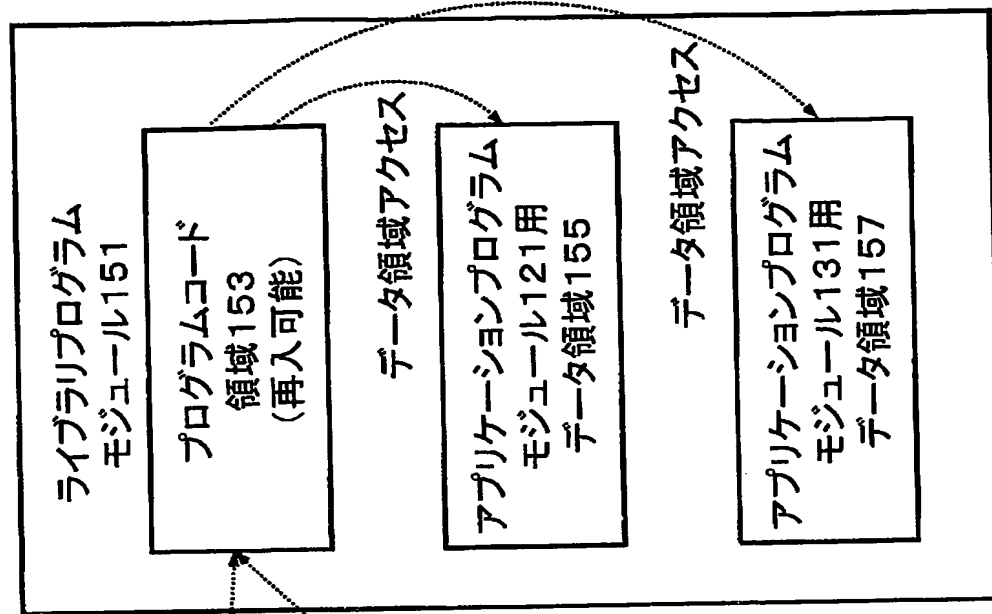
559 共用データ領域

【書類名】

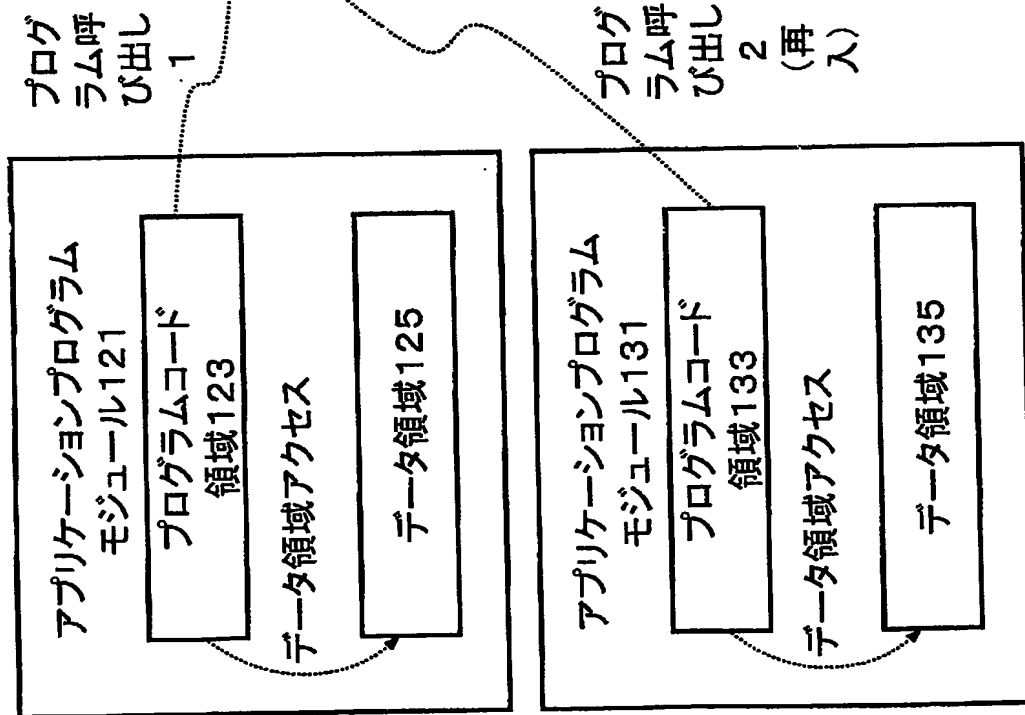
図面

【図1】

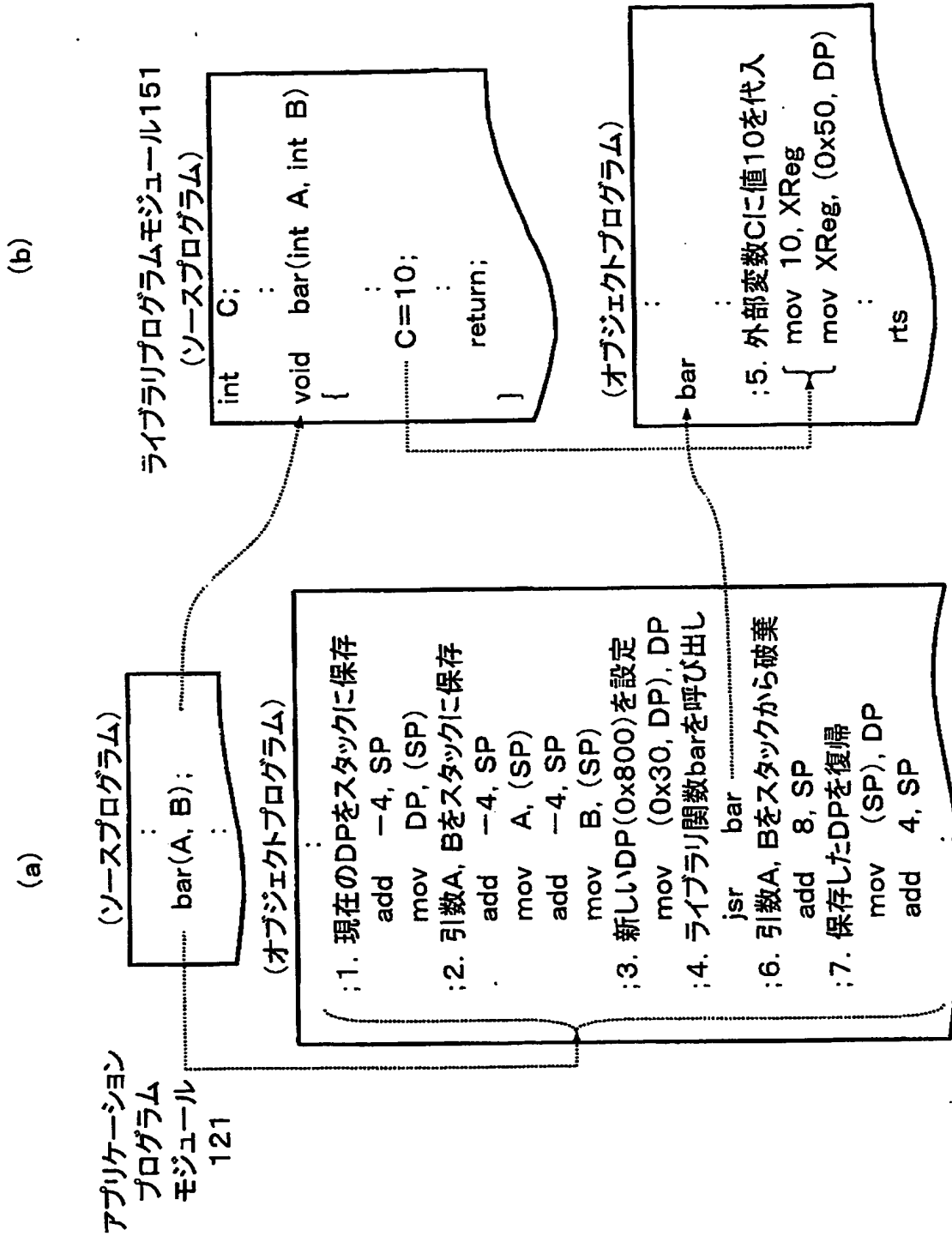
(b)



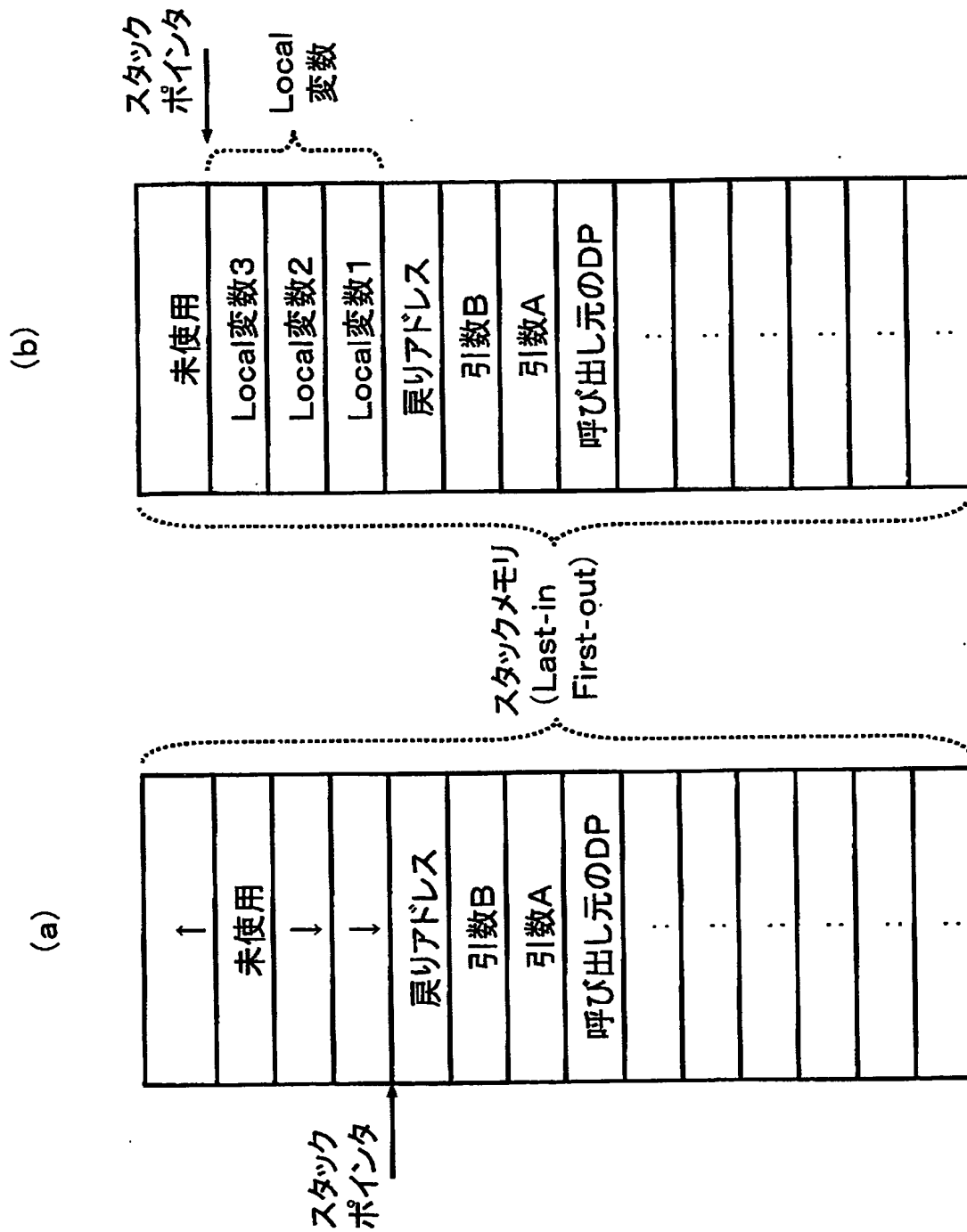
(a)



【図3】



【図 4】



【図6】

(a)

ライブラリプログラムモジュール551
(ソースプログラム)

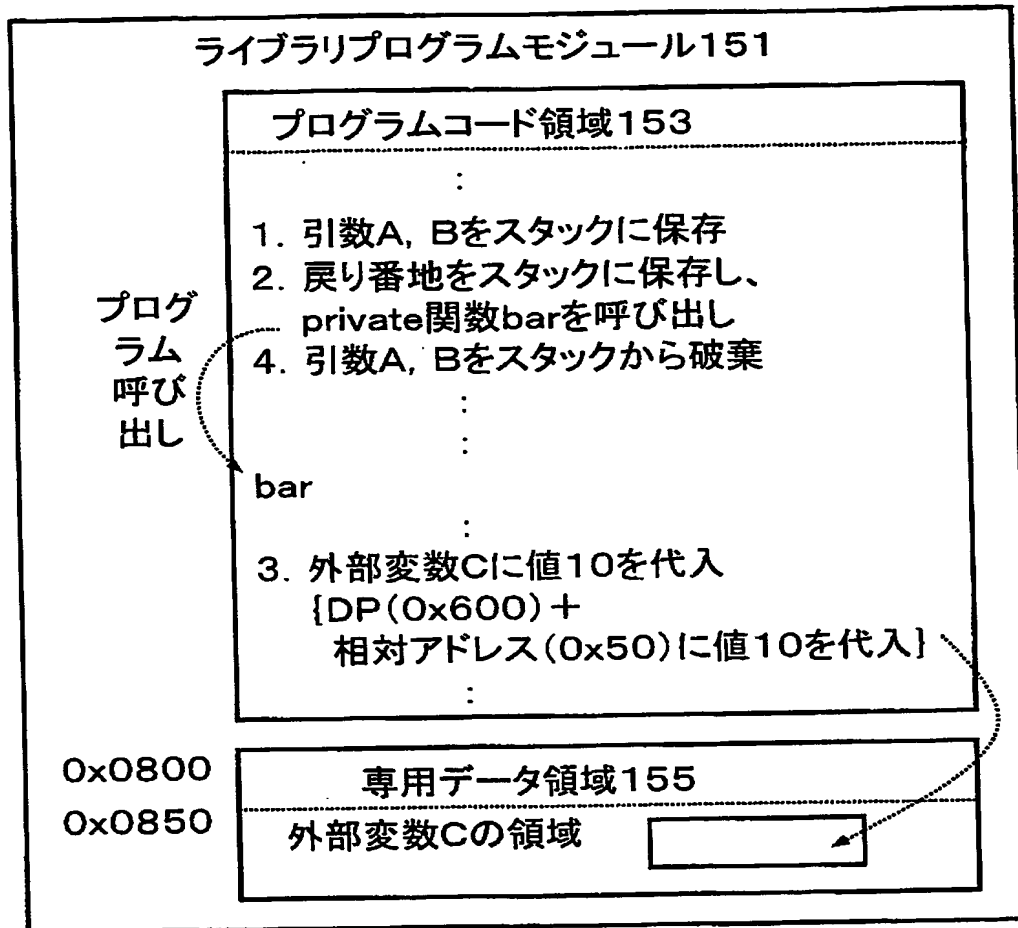
```
int C;
common int D;
void bar(int A, int B)
{
    /* 外部変数Cに値10を代入 */
    C=10;
    /* 外部変数Dに値20を代入 */
    D=20;
    return;
}
```

(b)

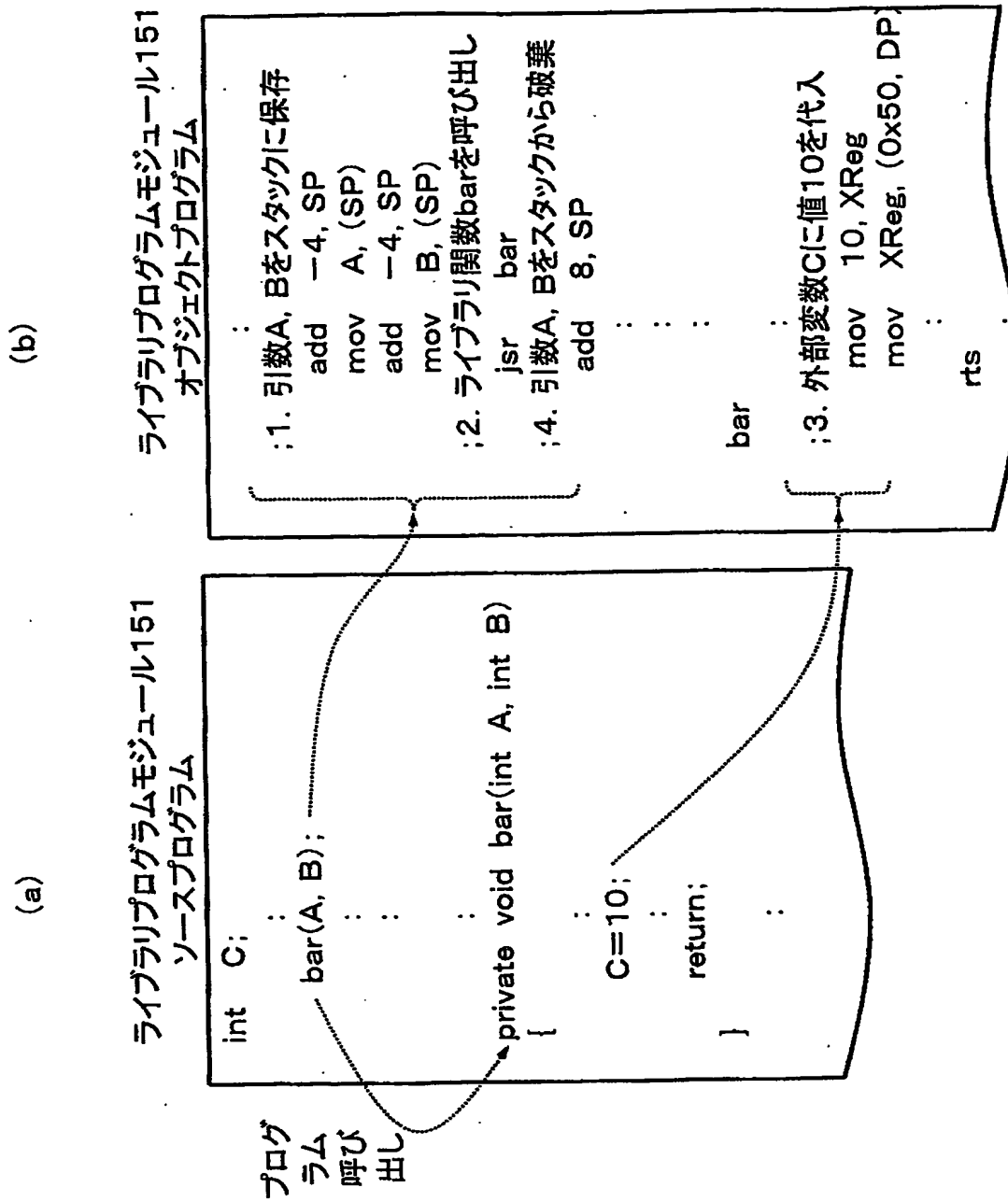
ライブラリプログラムモジュール551
(オブジェクトプログラム)

```
bar
:
:
; 外部変数Cに値10を代入
{ mov 10, XReg
; 外部変数Dに値20を代入
{ mov XReg, (0x50, DP)
{ mov 20, XReg
{ mov XReg, (0x60, SDP)
:
rts
:
```

【図7】

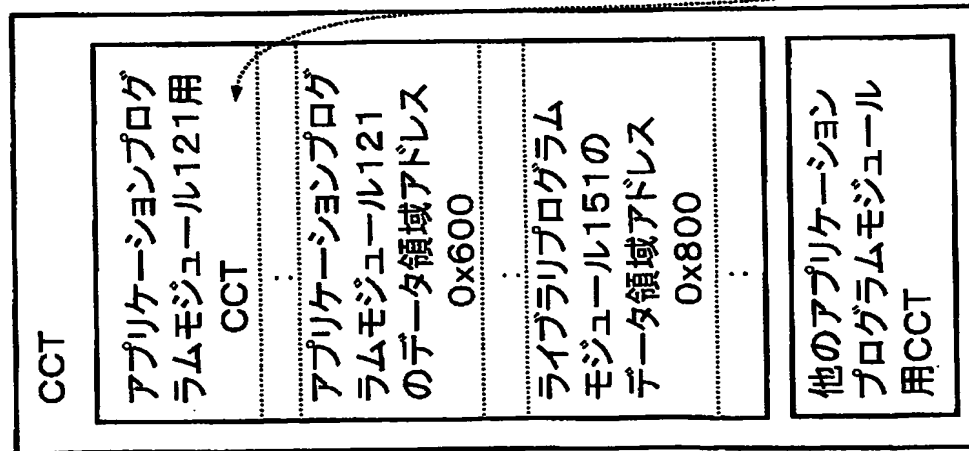


【图 8】

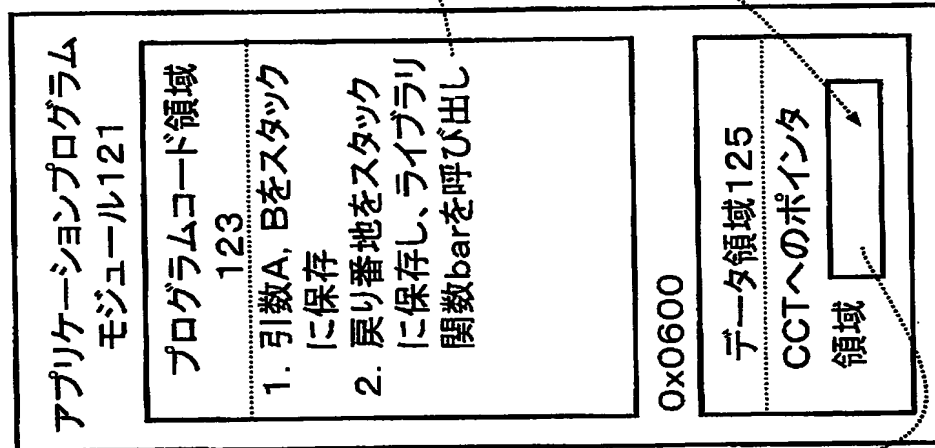


【図9】

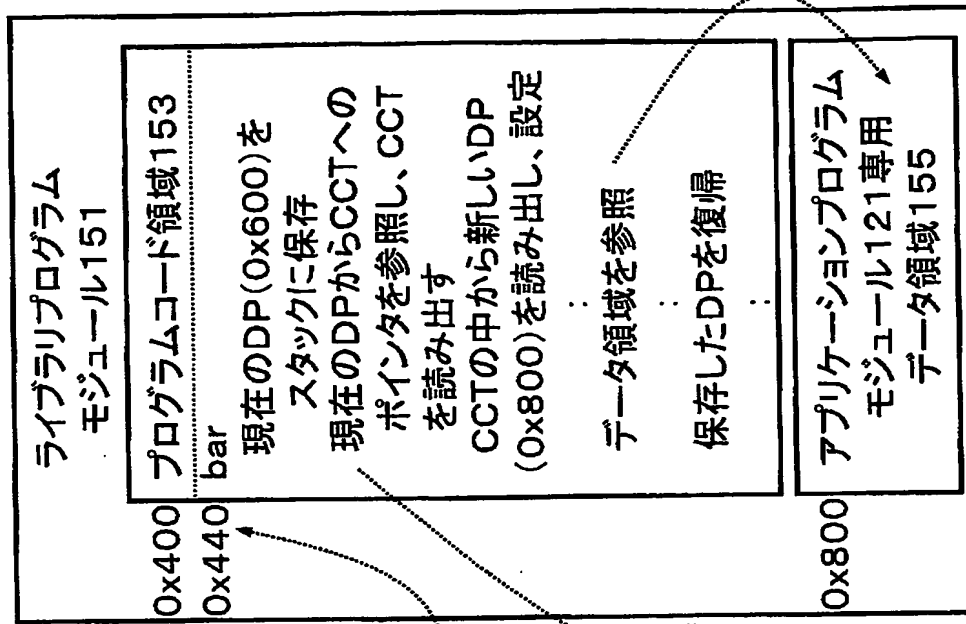
(a)



(b)



(c)



【書類名】 要約書

【要約】

【課題】 再入可能なプログラムの作成は、その本質的部分を人間のプログラマの作業によって記述されるため、多くの時間を要しながら不完全な排他的制御しか実現することができず、その結果、プログラムを実行させた時に多くの誤動作を引き起こす原因となっていた。

【解決手段】 呼び出し元プログラムモジュールから他プログラムモジュールを呼び出すステップを前記呼び出される他プログラムモジュール用データメモリ領域アドレスを設定するステップを含む目的プログラムのステップに変換し、呼び出される他プログラムモジュールに含まれるステップを前記設定された呼び出される他プログラム用データメモリ領域アドレスを参照してデータメモリ領域アクセスするステップを含む目的プログラムのステップに変換する。

【選択図】 図1

特願 2 0 0 3 - 0 0 0 2 7 5

出 願 人 履 歴 情 報

識別番号

[0 0 0 0 0 5 8 2 1]

1. 変更年月日

1 9 9 0 年 8 月 2 8 日

[変更理由]

新規登録

住 所

大阪府門真市大字門真 1 0 0 6 番地

氏 名

松下電器産業株式会社